US 2003/0084296A1

(54) **ACCESS PRIVILEGE AUTHENTICATION OF CLIENT COMPUTER FOR SERVICES PROVIDED BY SEVER COMPUTER**

(76) Inventors: **Masaki Kyojima**, Tokyo (JP); **Koji Takeda**, Tokyo (JP)

Correspondence Address:
NOTARO & MICHALOS P.C.
Suite No. 110
100 Dutch Hill Road
Orangeburg, NY 10962-2100 (US)

(57) **ABSTRACT**

Access privileges of a client are authenticated without using user account data. If an RSA modulus and public key have been assigned to the requested web page, the access privilege authentication controller 105 calls the challenge generator 107 to generate a challenge and transmits this challenge to the client 201 via the I/O controller 102. Subsequently, the access privilege authenticator 104 waits for a response from the client 201 sent via the I/O controller 102. When a response is received from the client 201, the access privilege authentication controller 105 calls the access privilege verifier 108 to verify whether the response is correct. If the response is correct, data indicating that the access privileges have been successfully authenticated is outputted, thereby alloying the client to access the requested service.

FIG.1

WEB SERVER

PAGE DATA
CONTROLLER
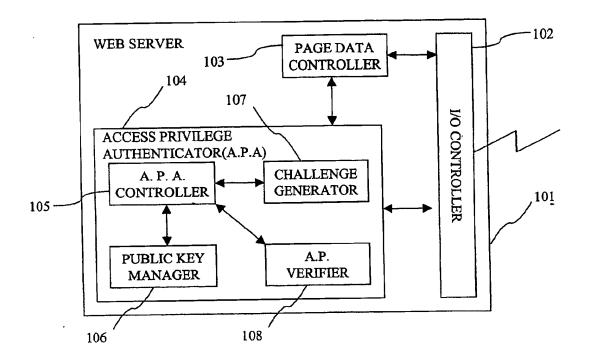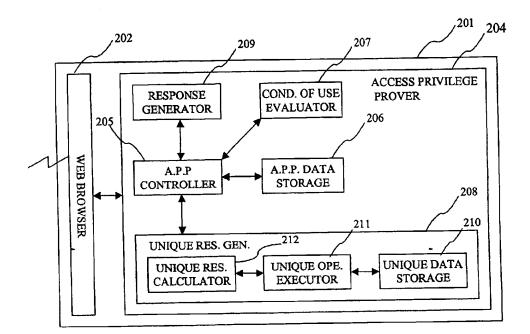
103

102

104

107

ACCESS PRIVILEGE
AUTHENTICATOR(A.P.A)

A. P. A.
CONTROLLER

CHALLENGE
GENERATOR

105

I/O CONTROLLER

101

PUBLIC KEY
MANAGER

A.P.
VERIFIER

106

108

FIG.2

FIG.3

```
                    ┌──────────────┐
                    │    Start     │
                    └──────┬───────┘
                           │
                           ▼                        301
            ┌──────────────────────────────┐
            │ Search for modulus/public    │
            │ key assigned to web page     │
            └──────────────┬───────────────┘
                           │
                           ▼                        302
                      ╱─────────╲                   NO
                  ╱─────────────────╲
                ╱  Have modulus/public ╲───────────────┐
                ╲  key been assigned to ╱               │
                  ╲   web page?       ╱                 │
                      ╲─────────╱                       │
                           │ YES                        │
                           ▼                   303      │
            ┌──────────────────────────────┐            │
            │ Generate challenge           │            │
            │ and send to client           │            │
            └──────────────┬───────────────┘            │
                           │                            │
                           ▼                   304      │
        NO            ╱─────────╲                        │
     ┌───────────────╱ Response  ╲                       │
     │               ╲ received  ╱                       │
     │                ╲from client╱                      │
     │                   ╲─────╱                          │
     │                     │ YES                305       │
     │                     ▼                              │
     │        ┌───────────────────────┐                  │
     │        │   Check response      │                  │
     │        └───────────┬───────────┘                  │
     │                    │                     306       │
     │      NO       ╱─────────╲                          │
     ├──────────────╱ Is response╲                        │
     │              ╲  correct?  ╱                        │
     │               ╲─────────╱                          │
     │                    │ YES          307              │
     │     308            ▼◄─────────────────────────────┘
 ┌──────────────────┐ ┌──────────────────────┐
 │ Output data      │ │ Output data          │
 │ indicating       │ │ indicating           │
 │ failed           │ │ successful           │
 │ verification     │ │ verification         │
 └────────┬─────────┘ └──────────┬───────────┘
          │                      │
          └──────────►───────────┤
                                 ▼
                          ┌─────────────┐
                          │     End     │
                          └─────────────┘
```

FIG.4

```
            ┌──────────┐
            │  Start   │
            └────┬─────┘
                 │                        401
                 ▼
        ┌─────────────────────┐
        │ Search for valid access │
        │ privilege proving data  │
        └──────────┬──────────┘
                   │                      402
                   ▼
              ╱──────────╲              NO
            ╱  Does valid  ╲─────────────────┐
          ╱ access privilege ╲               │
          ╲ proving          ╱               │
            ╲ data exist?   ╱                │
              ╲──────────╱                   │
                   │ YES          403        │
                   ▼                         │
        ┌─────────────────────┐              │
        │ Check conditions listed │          │
        │ in condition-of-use data│          │
        └──────────┬──────────┘              │
                   │            404          │
                   ▼                         │
              ╱──────────╲     NO            │
            ╱ Condition is  ╲────────────────┤
            ╲   met ?      ╱                 │
              ╲──────────╱                   │
                   │ YES          405        │
                   ▼                         │
        ┌─────────────────────┐              │
        │ Calculate response      │          │
        │ and send to web page    │          │
        └──────────┬──────────┘              │
                   │                         │
                   ▼◄────────────────────────┘
            ┌──────────┐
            │   End    │
            └──────────┘
```

FIG.5

FIG.6

FIG.7

701

704

702

707

ACCESS PRIVILEGE PROVER

| COND. OF USE EVALUATOR |

705

| A.P.P. CONTROLLER | ⟷ | A.P.P. DATA STORAGE |

706

WEB BROWSER

RESPONSE GENERATOR — 711

708

709

710

| RESPONSE CALCULATOR | ⟷ | UNIQUE OPE. EXECUTOR | ⟷ | UNIQUE DATA STORAGE |

FIG.8

```
                    ┌─────────────┐
                    │    Start    │
                    └─────────────┘
                           │                          801
                           ▼
                ┌────────────────────────┐
                │  Generate challenge    │
                │  and send to client    │
                └────────────────────────┘
                           │                          802
                           ▼
        NO    ◇─────────────────────────◇
      ◄───────   Response received
                    from client?
               ◇─────────────────────────◇
       │                   │ YES                      803
       │                   ▼
       │        ┌────────────────────────┐
       │        │     Check response     │
       │        └────────────────────────┘
       │                   │                          804
       │                   ▼
       │      NO  ◇──────────────────◇
       │     ◄─────   Is response
       │                 correct?
       │          ◇──────────────────◇
       │  806              │ YES           805
       │   │               ▼
  ┌─────────────────┐  ┌──────────────────┐
  │ Output data     │  │ Output data      │
  │ indicating      │  │ indicating       │
  │ failed          │  │ successful       │
  │ verification    │  │ verification     │
  └─────────────────┘  └──────────────────┘
       │                   │
       └───────────┬───────┘
                   ▼
            ┌─────────────┐
            │     End     │
            └─────────────┘
```

FIG.9

```
            ┌──────────┐
            │  Start   │
            └──────────┘
                 │
                 ▼
        ┌────────────────────┐  901
        │ Search for valid access
        │ privilege proving data │
        └────────────────────┘
                 │
                 ▼
             ╱────────╲  902
            ╱ Does valid ╲    NO
           ╱ access privilege proving ╲──────────┐
            ╲ data exist? ╱                      │
             ╲────────╱                          │
                 │ YES                           │
                 ▼         903                    │
        ┌────────────────────┐                   │
        │ Check conditions listed                │
        │ in condition-of-use data │             │
        └────────────────────┘                   │
                 │                                │
                 ▼        904                     │
             ╱────────╲                           │
            ╱ Condition is met? ╲  NO             │
             ╲────────╱──────────────────────────┤
                 │ YES                            │
                 ▼         905                    │
        ┌────────────────────┐                   │
        │ Calculate response and send            │
        │ to web server along with               │
        │ access privilege proving data │        │
        └────────────────────┘                   │
                 │◄──────────────────────────────┘
                 ▼
            ┌──────────┐
            │   End    │
            └──────────┘
```

FIG.10

```
701
702                707                     704

        ┌──────────────────────────────────────┐
        │  ┌─────────────┐     ACCESS           │
        │  │ COND. OF USE │    PRIVILEGE         │
        │  │  EVALUATOR   │    PROVER            │
   705  │  └─────────────┘                       │
W       │         ↕                              │
E       │  ┌─────────────┐                       │
B   ←──→│  │   A.P.P.     │                       │
        │  │  CONTROLLER  │                       │
B       │  └─────────────┘                       │
R       │         ↕                              │
O       │                          1001          │
W       │  ┌─────────────┐                       │
S       │  │ SMART CARD   │                       │
E       │  │  CONNECTOR   │                       │
R       │  └─────────────┘                       │
        └──────────────────────────────────────┘

                      1003           706              1002

        ┌──────────────────────────────────────────────┐
        │  ┌─────────────┐   ┌────────────┐  SMART CARD  │
        │  │I/O CONTROLLER│←→│ A.P.P. DATA │              │
        │  └─────────────┘   │  STORAGE   │              │
        │         ↕          └────────────┘         708  │
        │                                           709  │
        │ RESPONSE GENERATOR  711    710                 │
        │  ┌──────────┐ ┌──────────┐ ┌──────────────┐   │
        │  │ RESPONSE  │←│UNIQUE OPE.│←│ UNIQUE DATA  │   │
        │  │CALCULATOR │→│ EXECUTOR  │→│   STORAGE    │   │
        │  └──────────┘ └──────────┘ └──────────────┘   │
        └──────────────────────────────────────────────┘
```

ACCESS PRIVILEGE AUTHENTICATION OF CLIENT COMPUTER FOR SERVICES PROVIDED BY SEVER COMPUTER

BACKGROUND OF THE INVENTION

[0001] 1. Field of the Invention

[0002] The present invention relates to an authentication technology enabling a server to verify whether a client connected to the server via a network has valid authorization to access services on the server.

[0003] 2. Description of the Related Art

[0004] Most servers providing services to client computers that are connected there to via a network have a process for determining, prior to providing these services, whether the client has proper access privileges for the services or whether the client computer is currently being operated by a user having proper access privileges for the services. This process is called client authentication.

[0005] Pages 353-359 of "Microsoft Internet Information Server Resource Kit" issued by Microsoft Corporation of the U.S. describe client authentication functions possessed by a Microsoft Internet Information Server (herein after abbreviated to IIS. "Microsoft" is a trademark of Microsoft Corporation). The IIS has a system for client authentication called Basic Authentication. In Basic Authentication, an account is assigned to the user who accesses a server using a client. The accounts of the users that are authorized to access files stored on the server are determined.

[0006] When receiving a request from a client for a file on the server, the server requests the user to input a user ID and password. Based on the inputted user ID and password, the server identifies the user's account, determines whether the requested file can be accessed by that account, and provides the requested file to the client computer when access is authorized.

[0007] Authentication methods such as Basic Authentication in IIS are extremely common. In fact, nearly all applications using the Web, whether via the Internet or an intranet, employ similar authentication methods with Basic Authentication for restricting user access.

[0008] The following two problems exist in client authentication using user accounts, such as Basic Authentication in IIS.

[0009] (1) Management of user accounts places heavy maintenance and operation cost on the server

[0010] (2) User anonymity on the server end is not guaranteed

[0011] Next, these problems will be described in greater detail.

[0012] In client authorization using user accounts, it is essential that the server can identify the user's account. This means that the server must manage the user accounts. Accordingly, the server must have a database for managing the accounts. This database must be modified daily in response to the addition and deletion of accounts, resulting in considerable maintenance and operating costs.

[0013] When many servers are performing the same client authorization process, a main server (account server) is provided for managing the user accounts. Each server refers to the main server for account information. This method requires increased communication costs and security problems between individual servers and the account server.

[0014] In addition, users must reveal their accounts to the server. This means that the history of service usage by each user is exposed to the server. Considering the innumerable cases of abusing private data collected on servers in today's society, it is inevitable that this method poses a large risk for users.

SUMMARY OF THE INVENTION

[0015] In actual fact, identifying user accounts is not essential for client authentication. The purpose of performing client authentication is simply to determine whether it is permissible to provide a service to the client requesting that service or to the user of that client. Identifying the user account is nothing more than a means to make that determination, not the objective.

[0016] According to the present invention, public keys are assigned to services provided to the clients. In order to determine whether or not to supply a service to a client or to a user of that client, the server executes a challenge/response exchange using a public key assigned to the requested service.

[0017] For example, when using the RSA (Rivest, Shamir, and Adleman) algorithm for public-key encryption a modulus n and public key e of the RSA key pair (public key e and private key d) are assigned to a service, and the private key d is distributed to all clients authorized to receive that service. When a client requests a service that has been assigned a modulus n and public key e, the server generates a random number C, called a challenge, and sends this value C to the client. The client uses the private key d it possesses to calculate a response according to Equation (1) and returns the result to the server. Here, either the server can transmit the modulus n to the client along with the challenge, or the modulus n can be distributed to the client along with the private key d. Since the modulus n is public data, this value can also be stored on a separate server accessible by anyone.

$$R = C^d \bmod n \tag{1}$$

[0018] After receiving a response, the server verifies whether Equation (2) is satisfied using the values for the challenge C sent to the client, the response R received from the client, and the modulus n and public key e assigned to the service.

$$C = R^e \bmod n \tag{2}$$

[0019] If the relationship in Equation (2) holds, the service is provided to the client. If not, the service is not provided.

[0020] In this method, the private key d serves as access privilege proving data to prove the access privileges of a client for the service assigned the modulus n and public key e. All users having the private key d can access services assigned the modulus n and public key e.

[0021] The same private key can serve to enable use of a plurality of services assigned the same modulus and public key. These can be thought of as a group of services that can be used according to the same access privileges, whether they are on the same server or completely different servers.

Conversely, a plurality of services, each assigned a different modulus and public key, can be thought to require different access privileges.

[0022] With this method, the server need not identify the account of the user operating the client. A database for user accounts is unnecessary, and the server need not access an account server to identify a user account. Further, the user accounts are not made public to the server. In fact, in the challenge/response process, data identifying the user is never transmitted to the server side, thereby user's anonymity is completely protected.

[0023] RSA is not the only public-key encryption algorithm that can be used in the above method. It is possible to apply a challenge/response method using a signature, such as DSA (Digital Signature Algorithm). Another possible algorithm is one that requires commitment transmissions prior to the challenge/response change, such as Schnorr Signature Authentication Algorithm.

[0024] The problem with these methods is that the private key and modulus sent to the client as data for proving access privileges can be copied to another client, running the risk of the service being used dishonestly.

[0025] To solve these problems, the present invention assigns unique operations to each client. Access privilege proving data is created using this unique operation and the private key corresponding to the modulus and public key assigned to the desired service.

[0026] For example, access privilege proving data t is created as in Equation (3), where n is the modulus, e the public key, d the private key, and f_U ($\bullet$) denotes the unique operation possessed by the client U. The calculated value t is provided to the client U. Here, "|" denotes bit concatenation.

$$t = d - f_U(n|e) \tag{3}$$

[0027] When a request for a service assigned the modulus n and public key e is received from the client U, the server generates a challenge C and transmits this value to the client U. The client U calculates a response based on Equation (4) and returns this value to the server. Here, the server can transmit the modulus n and public key e to the client along with the challenge. The modulus n and public key can also be provided to the client in advance together with the access privilege proving data t. Since it is public, the modulus n and public key e can also be stored on a separate server accessible by anyone.

$$R = C^t C^{f_U(n|e)} \tag{4}$$

[0028] After receiving a response, the server verifies whether Equation (2) is satisfied using the values for the challenge C sent to the client, the response R received from the client, and the modulus n and public key e assigned to the service. If the relationship in Equation (2) holds, the service is provided to the client. If not, the service is not provided.

[0029] This method achieves the same functions as the method described above because services are not provided to any computer other than those clients having access privilege proving data assigned to the services. An advantage of this method is that access privilege proving data for one service is different for each client. Therefore, this access privilege proving data cannot be used dishonestly, even if

the data is copied, providing the results of the unique operation $f_U(\bullet)$ are protected from accesses by the user of the client.

[0030] In the above-described method, the client performs the operation using access privilege proving data. However, the same effects can be achieved by configuring the server to use the access privilege proving data. The procedure used for this method is described next.

[0031] When a request for a service assigned the modulus n and public key e is received from the client U, the server generates a challenge C and transmits this value to the client U. The client U calculates a response based on Equation (5) and returns this value to the server along with the access privilege proving data t.

$$R = C^{f_U(n|e)} \tag{5}$$

[0032] After receiving a response, the server verifies whether Equation (6) is satisfied using the values for the challenge C sent to the client, the response R and the access privilege proving data t received from the client, and the modulus n and public key e assigned to the service. If the relationship in Equation (6) holds, the service is provided to the client. If not, the service is not provided.

$$C = C^t R \bmod n \tag{6}$$

[0033] One method for generating the unique operation is to assign a common cryptographic hash function f($\bullet$) to all clients (for example, SHA-1) and to assign a different data u for each client, then performing the operation shown in Equation (7).

$$f_U(x) = f(x|u) \tag{7}$$

[0034] It is also possible to use the client-specific data u itself as the result of the unique operation.

[0035] When it is desirable that the use of services provided by the server be limited by specified conditions, data describing usage conditions (conditions-of-use data) is used. For example, when restricting the time period in which the services can be used, the starting and ending times of use are described in the conditions-of-use data.

[0036] The access privilege proving data is then generated from a private key, a unique client-specific operation, and the conditions-of-use data. For example, if d is the private key, $f_U(\bullet)$ the unique operation, and L the conditions-of-use data, the access privilege proving data t is calculated as shown in Equation (8).

$$t = d - f_U(n|e|L) \tag{8}$$

[0037] The access privilege proving data t and conditions-of-use data L are provided to the client in advance.

[0038] When a request for a service assigned the modulus n and public key e is received from the client U, the server generates a challenge C and transmits this value to the client U. First, the client evaluates the conditions of use described by L. If those conditions have been met, the client calculates a response based on Equation (9) and returns this value to the server.

$$R = C^t C^{f_U(n|e|L)} \tag{9}$$

[0039] After receiving a response, the server verifies whether Equation (2) is satisfied using the values for the challenge C sent to the client, the response R received from the client, and the modulus n and public key e assigned to

the service. If the relationship in Equation (2) holds, the service is provided to the client. If not, the service is not provided.

[0040] Using this method, the value of the response R changes when the conditions of use described by L are modified. As a result, Equation (2) is not satisfied. Accordingly, the user of the client can no longer over write L for the user's own benefit.

[0041] The methods described thus far assign a unique operation to the client. Since access privilege proving data is created based on this operation, different access privilege proving data is required for a different client, even when the user is the same. Further, it is difficult to control whether usage of services is permitted or not permitted among a plurality of users using the same client.

[0042] With such methods that assign unique operations to clients, it is difficult to implement access privilege control that truly binds to users. The present invention solves this type of problem by implementing a unique operation unit with a portable device, such as a smart card (an IC card with a calculating capacity) The user carries this portable device around and connects the device to the client to be used. This method achieves access privilege control that binds only to the user, regardless what client is being used.

[0043] The present invention can be implemented as a system or apparatus such as a server (server computer), a client, a computer system, and also implemented as a method. Obviously, at least a portion of this system can be implemented with a computer program. The client can be configured of a variety of data terminals, including a personal computer, workstation, portable terminal, set top box, home electric data appliance, and intelligent telephone (including portable telephones). The server can be a web server, FTP server, application server, or other server providing a variety of services.

[0044] The server and clients can be connected via a wired or wireless network employing such configurations as an IP network, a telephone network, or a mobile communication network. The Internet, intranets, extranets, and the like can be applied to suit various environments of use. The system can also employ communication channels using short-range radio waves, as in the Bluetooth (trademark) method. A device using short-range radio waves can be used as a client in this case.

[0045] A detailed description has been given above for the present invention, various aspects of which are described in the Claims. Next, preferred embodiments of the present invention will be described in detail.

BRIEF DESCRIPTION OF THE DRAWINGS

[0046] In the drawings:

[0047] FIG. 1 is a block diagram showing the configuration of a web server according to a first embodiment of the present invention;

[0048] FIG. 2 is a block diagram showing the configuration of a client according to the first embodiment;

[0049] FIG. 3 is a flowchart illustrating the method of operations according to the first embodiment;

[0050] FIG. 4 is a flowchart illustrating the method of operations according to the first embodiment;

[0051] FIG. 5 is a block diagram showing a variation of the first embodiment;

[0052] FIG. 6 is a block diagram showing the configuration of a web server according to a second embodiment of the present invention;

[0053] FIG. 7 is a block diagram showing the configuration of a client according to the second embodiment;

[0054] FIG. 8 is a flowchart illustrating the method of operations according to the second embodiment;

[0055] FIG. 9 is a flowchart illustrating the method of operations according to the second embodiment; and

[0056] FIG. 10 is a block diagram showing a variation of the second embodiment.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0057] An authentication technology according to preferred embodiments of the present invention will be described while referring to the accompanying drawings.

[0058] First Embodiment

[0059] The authentication technology of the first embodiment comprises a web server for providing web pages to clients and clients that receive web pages after requesting the web pages from the web server.

[0060] The web server of the present embodiment has an access privilege authentication mechanism applying the present invention. An RSA modulus and a public key are assigned to web pages on the web server when necessary. Web pages not assigned with these values are accessible by any client.

[0061] When a request is received from a client for a web page assigned an RSA modulus and public key, the web server executes a challenge/response exchange with the client to verify whether the client has access privileges for that page. The challenge/response is successful only when the client possesses access privilege proving data corresponding to the RSA modulus and public key that are assigned to the relevant web page. The requested web page is provided to the client only when this verification is successful.

[0062] The method of assigning the RSA modulus and public key values to web pages has a number of possible variations. For example, it is possible to associate an RSA modulus and public key to each web page. It is also possible to associate an RSA modulus and public key to a group of web pages listed in a directory or the like. When a request for any web page contained in this group of web pages is received, a challenge/response exchange is executed using the RSA modulus and public key that binds this group.

[0063] The clients of the present embodiment possess an access privilege proving mechanism applying the present invention. When a client receives a challenge from a web server in response to a web page request initiated by the client, the client can generate and return a response to the challenge.

[0064] The clients in the present embodiment are each assigned unique data. Access privilege proving data giving specific clients access to a certain web page is created from a private key corresponding to the RSA modulus and public key assigned to that web page, the unique data assigned to the client, and conditions-of-use data describing conditions in which use of the web page is permitted to clients. The conditions-of-use data includes a description of the starting and ending time and date for a period in which the client can access the web page.

[0065] As an example, let us say n and e are the RSA modulus and the public key assigned to the web page. d is the private key corresponding to n and e. u is the unique data assigned to the client, and L is the conditions-of-use data. In this example, the access privilege proving data t is calculated according to Equation (10) below.

$$t = d - f(n|e|L|u) \tag{10}$$

[0066] Here, f(●) is a cryptographic hash function universally possessed by all clients.

[0067] FIG. 1 shows the internal construction of a web server 101 according to the first embodiment of the present invention. The web server 101 includes an input/output (I/O) controller 102, a page data manager 103, and an access privilege authenticator 104.

[0068] The I/O controller 102 controls input received by the web server 101 from a client via a network and output transmitted from the web server 101 to a client via the network.

[0069] The page data manager 103 manages web pages on the web server. The page data manager 103 includes a function for supplying a web page via the I/O controller 102 in response to a request for the page from a client. When necessary, the page data manager 103 calls the access privilege authenticator 104 before supplying a web page in order to confirm whether the client has proper access privileges and can cancel provision of the web page when the client does not.

[0070] The access privilege authenticator 104 determines whether a client has proper access privileges in response to a call from the page data manager 103 and returns the result to the page data manager 103. During the determination process, the access privilege authenticator 104 communicates with the client via the I/O controller 102.

[0071] The access privilege authenticator 104 comprises an access privilege authentication controller 105, a public key manager 106, a challenge generator 107, and an access privilege verifier 108.

[0072] The access privilege authentication controller 105 controls the overall process for testing whether the client has proper user privileges.

[0073] The public key manager 106 stores and manages the bindings between the web pages and their corresponding assigned RSA modulus and public key.

[0074] The legitimacy of the client's access privileges is determined according to a challenge/response exchange conducted between the web server and the client. The challenge generator 107 generates a challenge that is transmitted to the client.

[0075] The access privilege verifier 108 determines whether the response received from the client has a prescribed relationship with the challenge generated by the challenge generator 107 and the RSA modulus and public key assigned to the web page that has been requested by the client. After this determination, the access privilege verifier 108 outputs the result of this determination.

[0076] FIG. 2 shows the internal configuration of a client 201 according to the first embodiment of the present invention. The client 201 includes a web browser 202 and an access privilege prover 204.

[0077] The web browser 202 controls input to the client received from the web server via a network and output transmitted from the client to the web server via the network. The web browser 202 also serves to transmit requests for web pages to the web server and to process web pages received from the web server.

[0078] The access privilege provor 204 proves for the web server that the client has access privileges for a certain web page. When a challenge is received from the web server via the web browser 202 for verifying access privileges, the access privilege prover 204 calculates a response for proving its own access privileges and sends this response to the web server via the web browser 202.

[0079] The access privilege prover 204 comprises an access privilege proving controller 205, an access privilege proving data storage unit 206, a conditions-of-use evaluator 207, a unique response generator 208, and a response generator 209.

[0080] The access privilege proving controller 205 controls overall calculations of the response for proving access privileges of the client.

[0081] The access privilege proving data storage unit 206 stores access privilege proving data for web pages and is capable of.storing multiple access privilege proving data. Each separate access privilege proving data is stored along with its RSA modulus, public key, and conditions-of-use data used in generating the access privilege proving data.

[0082] The conditions-of-use evaluator 207 determines whether or not the conditions of use described by the conditions-of-use data are met. The conditions-of-use data in the present embodiment describe the starting and ending time and date for a period in which the client can access the web page. The conditions-of-use evaluator 207 includes a function for determining whether the current time falls within the period during which the web page is accessible.

[0083] In the present embodiment, the generation of a response is divided into two stages: a calculation using unique data and a calculation using access privilege proving data. The unique response generator 208 executes the calculation using unique data.

[0084] The response generator 209 generates a response to be transmitted to the web server 101.

[0085] The unique response generator 208 comprises a unique data storage unit 210, a unique operation executor 211, and a unique response calculator 212.

[0086] The unique data storage unit 210 stores unique data assigned to the client.

[0087] The unique operation executor 211 is provided with a built-in cryptographic hash function f(●) possessed universally by all clients. The unique operation executor 211 executes this hash operation using the unique data stored in the unique data storage unit 210.

[0088] The unique response calculator 212 executes one stage of the response calculation for the response to be sent to the web server 101 using the results from the unique operation executor 211.

[0089] When the web server 101 receives a request for a web page from the client 201, the web server 101 transfers the request to the page data manager 103 via the I/O controller 102. The page data manager 103 calls the access privilege authenticator 104 in order to confirm that the client 201 has access privileges for the requested web page. When calling the access privilege authenticator 104, data such as a file name is transferred to the access privilege authenticator 104 in order to identify the requested web page.

[0090] The access privilege authenticator 104 determines whether the client 201 making the request possesses access privileges for the requested web page and transmits the results of this determination to the page data manager 103. If it is confirmed that the client has proper access privileges, the web page is transmitted to the client 201. If it is determined otherwise, an error message is returned to the client 201.

[0091] FIG. 3 is a flowchart showing the operations performed in the access privilege authenticator 104 when called by the page data manager 103. The procedure for verifying access privileges executed by the access privilege authenticator 104 will be described with reference to this flowchart.

[0092] The access privilege authentication controller 105 controls operations of the access privilege authenticator 104. The access privilege authentication controller 105 first calls the public key manager 106 in step 301 to search for an RSA modulus and public key assigned to the web page requested by the client 201. If an assigned RSA modulus and public key do not exist (No in 302), data indicating that access privileges have been successfully authenticated is output in step 307 and the process ends.

[0093] If an RSA modulus and public key have been assigned to the requested web page (Yes in 302), the access privilege authentication controller 105 calls the challenge generator 107 to generate a challenge and transmits this challenge to the client 201 via the I/O controller 102 in step 303. The access privilege authentication controller 105 also transmits the RSA modulus and public key assigned to the requested web page to the client 201 along with the challenge.

[0094] Subsequently, the access privilege authenticator 104 waits for a response from the client 201 via the I/O controller 102. If no response has been received after a fixed interval has passed since the challenge was transmitted (No in 304), data indicating that authentication of access privileges has failed is output in step 308 and the process ends.

[0095] When a response is received from the client 201 (Yes in 304), the access privilege authentication controller 105 calls the access privilege verifier 108 in step 305 to verify whether the response is correct. Here, the access privilege verifier 108 verifies whether Equation (2) is satisfied when the challenge sent to the client 201 in step 303 is C, the response received in step 304 is R, the RSA modulus assigned to the requested web page is n, and the public key is e. If Equation (2) is satisfied (Yes in 306), data indicating that access privileges have been successfully authenticated is output in step 307, and the process ends. If Equation (2) is not satisfied (No in 306), data indicating that verification of access privileges has failed is output in step 308, and the process ends.

[0096] Meanwhile, having received a challenge, RSA modulus, and public key from the web server 101 via the web browser 202, the client 201 transfers this data to the access privilege prover 204 to generate a response, then sends the response to the web server 101 via the web browser 202.

[0097] FIG. 4 is a flowchart showing operations of the access privilege prover 204 to which the challenge, RSA modulus, and public key are transferred. The procedure of generating a response that is executed by the access privilege prover 204 will be described with reference to this flowchart.

[0098] The access privilege proving controller 205 controls the operations of the access privilege prover 204. In step 401, the access privilege proving controller 205 searches the access privilege proving data storage unit 206 for the conditions-of-use data and access privilege proving data corresponding to the RSA modulus and public key received from the web server 101. A plurality of access privilege proving data with the RSA modulus, public key, and conditions-of-use data used in generating the access privilege proving data is stored in the access privilege proving data storage unit 206. In this step, the access privilege proving controller 205 searches for access privilege proving data and conditions-of-use data bound to the same RSA modulus and public key as that received from the web server 101.

[0099] If no access privilege proving data and conditions-of-use data corresponding to the data received from the web server 101 exists (No in 402), the process ends. In this case, a response is not returned to the web server 101.

[0100] If access privilege proving data and conditions-of-use data corresponding to the data received from the web server 101 exists (Yes in 402), the conditions-of-use evaluator 207 is called to determine in step 403 whether the conditions of use listed in the conditions-of-use data found in step 401 are satisfied.

[0101] If the conditions of use are not satisfied (No in 404), the process ends. In this case, a response is not returned to the web server 101. If the conditions of use are satisfied (Yes in 404), a response is calculated and transmitted in step 405 to the web server 101 via the web browser 202, and the process ends.

[0102] In step 405, the unique response generator 208 is called to calculate the response based on unique data assigned to the client. When calling the unique response generator 208, the challenge, RSA modulus, and public key transferred from the web server 101 and the conditions-of-use data found in step 401 are transferred to the unique response generator 208.

[0103] In the unique response generator 208, the unique response calculator 212 calls the unique operation executor 211 to obtain calculation results based on unique data stored in the unique data storage unit 210. The unique operation executor 211 performs a prescribed operation based on the unique data stored in the unique data storage unit 210, the RSA modulus and public key received from the web server 101, and the conditions-of-use data found in step 401. The unique operation executor 211 performs the calculation shown in Equation (11), where n is the RSA modulus received from the web server 101, e is the public key, L is the conditions-of-use data found in step 401, and u is the unique data stored in the unique data storage unit 210 to obtain the result F.

$$F = f(n|e|L|u) \tag{11}$$

[0104] Subsequently, the unique response calculator 212 calculates R' using Equation (12) from the result calculated by the unique operation executor 211, and the challenge C and RSA modulus n transferred from the web server 101.

$$R' = C^F \bmod n \tag{12}$$

[0105] After receiving the result R' of the calculation from the unique response generator 208, the access privilege proving controller 205 calls the response generator 209 to calculate the response R. This response is generated by executing the calculation shown in Equation (13) from the challenge C and RSA modulus n received from the web server 101, the result R' received from the unique response generator 208, and the access privilege proving data t found in step 401.

$$R = C^t R' \bmod n \tag{13}$$

[0106] In the present embodiment, the response check in step 305 is successful if there is a correct combination of the RSA modulus n and public key e assigned to the requested web page, unique data u assigned to the client, and access privilege proving data t and conditions-of-use data L indicating access privileges of that client. As a result of the successful finding in step 305, data for the requested web page is transferred to the client. If any part of this data combination is different, the result in step 305 will not be successful, and the client will not be able to obtain data for the requested web page.

[0107] In the present embodiment, the access privilege proving data is defined by Equation (10). Accordingly, the ability of someone to analyze the contents of the unique response generator 208 would create a security problem. If the result of Equation (11) were revealed, a third party could calculate the private key d corresponding to the RSA modulus n and the public key e. A third party learning the private key d could use the system dishonestly. Even without knowledge of Equation (11), disclosing the unique data u stored in the unique data storage unit 210 could lead anyone to calculate this Equation (11) and to ultimately calculate the private key d. To prevent this problem, measures must be taken to avoid a third party analysis of the calculation process and the internal memory of the unique response generator 208. It is necessary to take such measures as configuring the unique response generator 208 using tamper-resistant hardware or using software technology to make the analysis of programs and memory difficult, for example.

[0108] Since smart cards (IC cards possessing operating functions) are highly tamper-resistant hardware, configuring the unique response generator 208 using smart cards is an effective method. In this case, unique data is assigned to the smart cards rather than to the client. By carrying the smart cards and connecting them to a client, the user can use any client and still exercise the same access privileges. This method is even more user-friendly if the storage unit 206 is included in the smart card.

[0109] FIG. 5 shows an example configuration of the client 201 when configuring the unique response generator 208 and access privilege proving data storage unit 206 with smart cards.

[0110] For convenience of this description, parts in FIG. 5 having the same function as those in FIG. 2 are designated by the same reference numerals, while new numbers are assigned to new parts not appearing in FIG. 2.

[0111] Unlike the configuration shown in FIG. 2, the unique response generator 208 and access privilege proving data storage unit 206 are included in a smart card 502 in the configuration of FIG. 5. The smart card 502 has an I/O controller 503 for controlling data input and output between the smart card 502 and client 201. The client 201 has a smart card connector 501 for connecting to the smart card 502 and controlling data input and output with the smart card. Since the remaining configuration in FIG. 5 is the same as that shown in FIG. 2, a description of these parts has been omitted.

[0112] Second Embodiment

[0113] The authentication technology according to a second embodiment of the present invention comprises a web server for providing web applications to clients, and clients using these web applications on the web server. Operations of the applications on the web server are controlled using script that is interpreted and executed on the server end.

[0114] The web server in the present embodiment employs a mechanism for authenticating access privileges according to the present invention. The timing at which the mechanism of access privilege authentication is called can be described in script, along with the RSA modulus and public key used by this access privilege authentication mechanism. During the process of interpreting and executing this script, the mechanism for access privilege authentication is called to verify that the client has access privileges. Before providing the client with specific functions of a web application, for example, it is possible to perform access privilege authentication for those functions in order to provide relevant functions only to clients confirmed to have proper access privileges.

[0115] When called, the mechanism of access privilege verification verifies whether the client has proper access privileges by performing a challenge/response exchange with the client. The challenge/response exchange is only successful when the client possesses access privilege proving data corresponding to an RSA modulus and public key specified for the access privilege authentication.

[0116] In the present embodiment, the client has a mechanism for authenticating access privileges according to the present invention. When the client receives a challenge from the web server, the client can generate and return a response. Each client in the present embodiment is assigned unique data. Access privilege proving data for special clients is

generated from the private key corresponding to the RSA modulus and public key, the unique data assigned to the client, and conditions-of-use data describing conditions in which use of functions on web application is allowed the client. The conditions-of-use data includes starting and ending times and dates for the valid time period of the client's access privileges.

[0117] For example, an access privilege proving data t is generated according to Equation (10), where n is the RSA modulus, e the public key, d the private key corresponding to the n and e, u the unique data assigned to the client, and L the conditions-of-use data.

[0118] FIG. 6 shows the internal construction of a web server 601 according to the second embodiment of the present invention. The web server 601 includes an I/O controller 602, a script interpreter 603, and an access privilege authenticator 604.

[0119] The I/O controller 602 controls input received by the web server 601 from a client via a network and output transmitted from the web server 601 to a client via the network.

[0120] The script interpreter 603 retains script configuring a web application that is provided by the web server 601, interprets and executes the script in response to a request from a client, and provides results to the client via the I/O controller 602. When interpreting and executing the script, the script interpreter 603 can call the access privilege authenticator 604 to verify whether the client has proper access privileges.

[0121] The access privilege authenticator 604 determines whether a client has proper access privileges in response to a call from the script interpreter 603 and returns the result to the script interpreter 603. During the determination process, the access privilege authenticator 604 communicates with the client via the I/O controller 602.

[0122] The access privilege authenticator 604 comprises an access privilege authentication controller 605, a challenge generator 606, and an access privilege verifier 607.

[0123] The access privilege authentication controller 605 controls the overall process for testing whether the client has proper user privileges.

[0124] The access privilege authentication control unit 605 determines whether the client has access privileges according to a challenge/response exchange conducted between the web server and the client. The challenge generator 606 generates a challenge that is transmitted to the client.

[0125] The access privilege verifier 607 determines whether the response received from the client has a pre-scribed relationship with the challenge generated by the challenge generator 606, the access privilege proving data transmitted by the client along with the response, and the RSA modulus and public key specified by the script interpreter 603, and outputs the result of this determination.

[0126] FIG. 7 shows the internal configuration of a client 701 according to the second embodiment of the present invention. The client 701 includes a web browser 702 and an access privilege prover 704.

[0127] The web browser 702 controls input to the client received from the web server via a network and output transmitted from the client to the web server via the network. The web browser 702 also serves to transmit requests for web applications to the web server and to process web pages received from the web server.

[0128] When a challenge is received from the web server 601 via the web browser 702 for authenticating access privileges, the access privilege prover 704 calculates a response for proving its own access privileges and sends this response to the web server 601 via the web browser 702, together with access privilege proving data.

[0129] The access privilege prover 704 comprises an access privilege proving controller 705, an access privilege proving data storage unit 706, a conditions-of-use evaluator 707, and a response generator 708.

[0130] The access privilege proving controller 705 controls overall calculations of the response for proving access privileges of the client.

[0131] The access privilege proving data storage unit 706 stores access privilege proving data possessed by the client and is capable of storing multiple access privilege proving data. Each separate access privilege proving data is stored along with its RSA modulus, public key, and conditions-of-use data used in generating access privilege proving data.

[0132] The conditions-of-use evaluator 707 determines whether or not the conditions of use described by the conditions-of-use data are met. The conditions-of-use data in the present embodiment describe the starting and ending time and date for the valid period of the corresponding access privilege proving data. The conditions-of-use evaluator 707 includes a function for determining whether the current time falls with in this period.

[0133] The response generator 708 generates a response to be transmitted to the web server 601.

[0134] The response generator 708 comprises a unique data storage unit 709, a unique operation executor 710, and a response calculator 711.

[0135] The unique data storage unit 709 stores unique data assigned to the client.

[0136] The unique operation executor 710 is provided with a built-in cryptographic hash function $f(\bullet)$ possessed universally by all clients. The unique operation executor 710 executes this hash operation using the unique data stored in the unique data storage unit 709.

[0137] The response calculator 711 calculates the response to be sent to the web server 601 using the results from the unique operation executor 710.

[0138] Part of the script retained by the web server 601 describes calling the mechanism for access privilege authentication. The RSA modulus and public key used for access privilege authentication are described in the script. When reaching the part of the script for calling the mechanism for access privilege authentication during the process of interpreting and executing the script, the script interpreter 603 calls the access privilege authenticator 604 for confirming whether the client 701 has access privileges corresponding to the RSA modulus and public key specified in the script.

[0139] The access privilege authenticator **604** verifies whether the client **701** has access privileges corresponding to the specified RSA modulus and public key and sends the result of this check to the script interpreter **603**.

[0140] FIG. 8 is a flowchart showing the operations performed in the access privilege authenticator **604** when called by the script interpreter **603**. The procedure for verifying access privileges executed by the access privilege authenticator **604** will be described with reference to this flowchart.

[0141] The access privilege authentication controller **605** controls operations of the access privilege authenticator **604**. The access privilege authentication controller **605** first calls the challenge generator **606**to generate a challenge and transmits this challenge to the client **701** via the I/O controller **602** in step **801**. The access privilege authentication controller **605** also transmits the RSA modulus and public key used for access privilege authentication to the client **701** along with the challenge.

[0142] Subsequently, the access privilege authenticator **604** waits for a response and access privilege proving data to be sent from the client **701** via the I/O controller **602**. If no response and access privilege proving data has been received when a fixed interval has passed since the challenge was transmitted (No in **802**), data indicating that verification of access privileges has failed is output in step **806**, and the process ends.

[0143] When a response and access privilege proving data is received from the client **701** (Yes in **802**), the access privilege authentication controller **605** calls the access privilege verifier **607** in step **803** to verify whether the response is correct. Here, the access privilege verifier **607** verifies whether Equation (6) is satisfied when the challenge sent to the client **701** in step **801** is C, the response received in step **802** is R, the access privilege proving data is t, and the RSA modulus used for access privilege authentication is n. If Equation (6) is satisfied (Yes in **804**), data indicating that access privileges have been successfully authenticated is output in step **805**, and the process ends. If Equation (6) is not satisfied (No in **804**), data indicating that authentication of access privileges has failed is output in step **806**, and the process ends.

[0144] Meanwhile, after receiving a challenge, RSA modulus, and public key from the web server **601** via the web browser **702**, the client **701** transfers this data to the access privilege prover **704** to generate a response, then sends the response to the web server **601** via the web browser **702**, together with the access privilege proving data corresponding to the RSA modulus and public key.

[0145] FIG. 9 is a flowchart showing operations of the access privilege prover **704**, which generates a response after receiving the challenge, RSA modulus, and public key. The procedure of generating a response that is executed by the access privilege prover **704** will be described with reference to this flowchart.

[0146] The access privilege proving controller **705** controls the operations of the access privilege prover **704**. In step **901**, the access privilege proving controller **705** searches the access privilege proving data storage unit **706** for the conditions-of-use data and access privilege proving data corresponding to the RSA modulus and public key

received from the web server **601**. A plurality of access privilege proving data bound to the RSA modulus, public key, and conditions-of-use data used in generating the access privilege proving data is stored in the access privilege proving data storage unit **706**. In this step, the access privilege proving controller **705** searches for access privilege proving data and conditions-of-use data bound to the same RSA modulus and public key as that received from the web server **601**.

[0147] If no access privilege proving data and conditions-of-use data corresponding to the data received from the web server **601** exists (No in **902**), the process ends. In this case, a response is not returned to the web server **601**.

[0148] If access privilege proving data and conditions-of-use data corresponding to the data received from the web server **601** exists (Yes in **902**), the conditions-of-use evaluator **707** is called to determine in step **903** whether the conditions of use listed in the conditions-of-use data found in step **901** are satisfied.

[0149] If the conditions of use are not satisfied (No in **904**), the process ends. In this case, a response is not returned to the web server **601**. If the conditions of use are satisfied (Yes in **904**), a response is calculated in step **905** and transmitted to the web server **601** via the web browser **702**, along with the access privilege proving data found in step **901**, and the process ends.

[0150] In step **905**, the response generator **708** is called to calculate the response based on unique data assigned to the client. When calling the response generator **708**, the challenge, RSA modulus, and public key received from the web server **601** are transferred to the response generator **708**.

[0151] In the response generator **708**, the response calculator **711** calls the unique operation executor **710** to obtain calculation results based on unique data stored in the unique data storage unit **709**. The unique operation executor **710** performs a prescribed operation based on the unique data stored in the unique data storage unit **709** and the RSA modulus and public key received from the web server **601**. The unique operation executor **710** performs the calculation shown in Equation (11) to obtain the result F, where n is the RSA modulus received from the web server **601**, e is the public key, L is the conditions-of-use data found in step **901**, and u is the unique data stored in the unique data storage unit **709**.

[0152] Subsequently, the response calculator **711** calculates the response R using Equation (14) from the result found by the unique operation executor **710**, and the challenge C and RSA modulus n transferred from the web server **601**.

$$R = C^t \bmod n \qquad (14)$$

[0153] In the present embodiment, the response check in step **803** is successful if there is a correct combination of the RSA modulus n and public key e specified by the script interpreter **603**, unique data u assigned to the client, and access privilege proving data t and conditions-of-use data L indicating access privileges of that client. If any part of this data combination is different, the result in step **803** will not be successful. The results of the verification are returned to the script interpreter **603**. Operations of the web application can be modified according to these results. For example, it

is possible to transmit different HTML (Hyper Text Markup Language) documents to the client when the access privilege authentication succeeds and when it fails. Further, various web languages, such as XML (Extensible Markup Language), can be used in place of HTML documents.

[0154] In the present embodiment, the access privilege proving data is defined by Equation (10). Accordingly, the ability to analyze the contents of the response generator 708 would create a security problem. If the result of Equation (11) were revealed, a third party could calculate the private key d corresponding to the RSA modulus n and the public key e. A third party learning the private key d could use the system dishonestly. Even without knowledge of Equation (11), disclosing of the unique data u stored in the unique data storage unit 709 could lead anyone to calculate this Equation (11) and to ultimately calculate the private key d. To prevent this problem, measures must be taken to avoid a third party analysis of the calculation process and the internal memory in the response generator 708. It is necessary to take such measures as configuring the response generator 708 using tamper-resistant hardware or using software technology to make the analysis of programs and memory difficult, for example.

[0155] As in the first embodiment, it is preferable to implement the response generator 708 using a smart card.

[0156] FIG. 10 shows an example configuration of the client 701 when configuring the response generator 708 and access privilege proving data storage unit 706 with a smart card.

[0157] For convenience of this description, parts in FIG. 10 having the same function as those in FIG. 7 are designated by the same reference numerals, while new numbers are assigned to new parts not appearing in FIG. 7.

[0158] Unlike the configuration shown in FIG. 7, the response generator 708 and access privilege proving data storage unit 706 are included in a smart card 1002 in the configuration of FIG. 10. The smart card 1002 has an I/O controller 1003 for controlling data input and output between the smart card 1002 and client 701. The client 701 has a smart card connector 1001 for connecting to the smart card 1002 and controlling data input and output with the smart card. Since the remaining configuration in FIG. 10 is the same as that shown in FIG. 7, a description of these parts has been omitted.

[0159] In the embodiments of the present invention described above, access privilege authentication is not performed using user account data. Therefore, there is no need to maintain user account data, thereby preventing the service history of users being known on the server end.

[0160] As described above, clients are authenticated without using user accounts in the present invention, thereby reducing the cost of maintaining user accounts and protecting user privacy.

[0161] While the invention has been described in detail with reference to specific embodiments thereof, it would be apparent to those skilled in the art that many modifications and variations may be made therein without departing from the spirit of the invention, the scope of which is defined by the attached claims.

What is claimed is:

1. A server that provides services to clients connected to the server via a network, the server comprising:

a public-key storage unit for storing public keys assigned to each service provided by the server;

a challenge generator for generating a challenge to be sent from the server to the client after the server receives a request for a service from the client;

an access privilege verifier for verifying, using a corresponding public key, whether a prescribed relationship exists between the challenge transmitted to the client and a response to that challenge received from the client; and

a controlling unit, while authenticating access privilege of the client for a service provided by the server,

for transmitting the challenge generated by the challenge generator to the client;

for receiving the response to that challenge returned from the client;

for verifying, with an access privilege verifier using a public key assigned to the service and stored in the public-key storage unit, whether a prescribed relationship exists between the challenge and the response; and

for providing the service to the client only when the access privilege verifier successfully verifies the relationship.

2. A client that requests services from a server connected to the client via a network, the client comprising:

a unique operation executor for executing a unique operation assigned to the client;

an access privilege proving data storage unit for storing access privilege proving data created from a private key corresponding to a public key assigned to the requested service and the result of the unique operation;

a response generator for generating a response to a challenge received from the server by executing a prescribed calculation using the result of the unique operation and the access privilege proving data; and

a controlling unit, while proving to the server that the client owns the access privilege for the service provided by the server,

for receiving the challenge from the server;

for generating, with the response generator, the response from (a) the challenge, (b) the result of the unique operation executed by the unique operation executor, and (c) the access privilege proving data stored in the access privilege proving data storage unit; and

for transmitting the created response to the server.

3. A client that requests services from a server connected to the client via a network, comprising:

a portable device connector for connecting to a portable device provided with a unique operation generator for executing unique operations;

an access privilege proving data storage unit for storing access privilege proving data created from a private key

corresponding to a public key assigned to the requested service and the result of the unique operation assigned to the portable device;

a response generator for generating a response to a challenge received from the server by executing a prescribed calculation using the result of the unique operation and the access privilege proving data; and

a controlling unit, while proving to the server that the client owns the access privilege for the service provided by the server,

for receiving the challenge from the server;

for generating, with the response generator, the response from (a) the challenge, (b) the result of the unique operation executed by the unique operation executor housed in the portable device connected to the portable device connector, and (c) the access privilege proving data stored in the access privilege proving data storage unit; and

for transmitting the created response to the server.

4. A server that provides services to clients connected to the server via a network, the server comprising:

a public-key storage unit for storing public keys assigned to each service provided by the server;

a challenge generator for generating a challenge to be sent from the server to the client after the server receives a request for a service from the client;

an access privilege verifier for verifying, using a corresponding public key, whether a prescribed relationship exists among the challenge transmitted to the client, a response to that challenge received from the client, and an access privilege proving data for proving the access privilege of the client; and

a controlling unit, while authenticating access privilege of the client for a service provided by the server,

for transmitting the challenge generated by the challenge generator to the client;

for receiving the response to that challenge returned from the client, and

for verifying, with an access privilege verifier using a public key assigned to the service and stored in the public-key storage unit, whether a prescribed relationship exists among the challenge, the access privilege proving data for proving the access privilege of the client for the service, and the response; and

for providing the service to the client only when the access privilege verifier successfully verifies the relationship.

5. A client that requests services from a server connected to the client via a network, the client comprising:

a unique operation executor for executing a unique operation assigned to the client;

an access privilege proving data storage unit for storing access privilege proving data created from a private key corresponding to a public key assigned to the requested service and the result of the unique operation;

a response generator for generating a response to a challenge received from the server by executing a prescribed calculation using the result of the unique operation; and

a controlling unit, while proving to the server that the client owns the access privilege for the service provided by the server,

for receiving the challenge from the server;

for generating, with the response generator, the response from (a) the challenge, and (b) the result of the unique operation executed by the unique operation executor; and

for transmitting to the server, the created response, and the access privilege proving data stored in the access privilege proving data storage unit.

6. A client that requests services from a server connected to the client via a network, comprising:

a portable device connector for connecting to a portable device provided with a unique operation generator for executing unique operations;

an access privilege proving data storage unit for storing access privilege proving data created from a private key corresponding to a public key assigned to the requested service and the result of the unique operation assigned to the portable device;

a response generator for generating a response to a challenge received from the server by executing a prescribed calculation using the result of the unique operation; and

a controlling unit, while proving to the server that the client owns the access privilege for the service provided by the server,

for receiving the challenge from the server;

for generating, with the response generator, the response from (a) the challenge, and (b) the result of the unique operation executed by the unique operation executor housed in the portable device connected to the portable device connector; and

for transmitting to the server, the created response, and the access privilege proving data stored in the access privilege proving data storage unit.

7. A client as recited in claim 3, wherein the access privilege proving data storage unit is included in the portable device connected to the portable device connector.

8. A client as recited in claim 6, wherein the access privilege proving data storage unit is included in the portable device connected to the portable device connector.

9. A server as recited in claim 1, wherein the server is a web server that supplies web applications to clients; and the public keys stored in the public-key storage unit are assigned to individual web pages provided to the clients and are used to verify the access privileges of the client for a web page when the server receives a request from the client to access a web page.

10. A server as recited in claim 4, wherein the server is a web server that supplies web applications to clients; and the public keys stored in the public-key storage unit are assigned to individual web pages provided to the clients and are used

to verify the access privileges of the client for a web page when the server receives a request from the client to access a web page.

11. A server as recited in claim 1, wherein the server is a web server that supplies web applications to clients; and the public keys stored in the public-key storage unit are as signed to groups of web pages provided to the clients and are used to verify the access privileges of the client for a web page when the server receives a request from the client to access a web page in one of the groups of web pages.

12. A server as recited in claim 4, wherein the server is a web server that supplies web applications to clients; and the public keys stored in the public-key storage unit are assigned to groups of web pages provided to the clients and are used to verify the access privileges of the client for a web page when the server receives a request from the client to access a web page in one of the groups of web pages.

13. A server that provides services to clients connected to the server via a network, the server comprising:

a script interpreter for interpreting script designed to control the contents of services that the server provides to clients and for controlling the operations of the server; and

a privilege authenticator for authenticating access privileges of the client when called by the script interpreter.

14. A server as recited in claim 13, wherein the privilege authenticator comprises:

a challenge generator for generating a challenge to be sent from the server to the client; and

an access privilege verifier that uses a public key to verify a prescribed relationship between the challenge transmitted to the client and a response to that challenge returned from the client;

and the privilege authenticator receives a public key for authenticating privileges of the client when called by the script interpreter, transmits the challenge generated by the challenge generator to the client, receives a response to the challenge sent by the client, and verifies, using the received public key, access privileges of the client by means of the access privilege verifier.

15. A server as recited in claim 13, wherein the privilege authenticator comprises:

a challenge generator for generating a challenge to be sent from the server to the client; and

an access privilege verifier that uses a public key to verify a prescribed relationship among the challenge transmitted to the client, a response to that challenge returned from the client, and access privilege proving data for verifying access privileges of the client;

and the privilege authenticator receives a public key for authenticating privileges of the client when called by the script interpreter, transmits the challenge generated by the challenge generator to the client, receives a response to the challenge sent by the client, and verifies, using the received public key, access privileges of the client by means of the access privilege verifier.

16. A method executed in a server for providing services from the server to clients connected to the server via a network after verifying the access privileges of the clients

for the services, public keys being assigned in advance to respective services provided by the server, the method comprising the steps of:

generating a challenge when a request for a service is received from a client and transmitting the challenge to the client;

receiving a response to the challenge returned from the client;

verifying that a prescribed relationship exists between the challenge sent to the client and the response received from the client using the public key as signed to the requested service; and

providing the requested service to the client only when the prescribed relationship exists.

17. A method executed in a client for proving its access privilege for a server when requesting a service from a server connected to the client via a network, the client being in advance assigned with a unique operation, the requested service being in advance assigned with a public key, the client in advance receiving access privilege proving data for expressing the access privilege of the client for the service, the access privilege proving data being created from a private key corresponding to a public key assigned to the service and the result of a unique operation assigned to the client, the method comprising the steps of:

receiving a challenge from the server;

executing the unique operation assigned thereto;

generating a response based on the challenge received from the server, the result of the unique operation, and the access privilege proving data; and

transmitting the response to the server.

18. A method executed in a server for providing services from the server to clients connected to the server via a network after verifying the access privileges of the clients for the services, public keys being assigned in advance to respective services provided by the server, the method comprising the steps of:

generating a challenge when a request for a service is received from a client and transmitting the challenge to the client;

receiving a response to the challenge from the client;

verifying that a prescribed relationship exists among the challenge sent to the client, the response received from the client, and the access privilege proving data indicative of access privilege of the client for the service, using the public key assigned to the requested service; and

providing the requested service to the client only when the prescribed relationship exists.

19. A method executed in a client for proving its access privilege for a server when requesting a service from a server connected to the client via a network, the client being in advance assigned with a unique operation, the requested service being in advance assigned with a public key, the client in advance receiving access privilege proving data for expressing the access privilege of the client for the service, the access privilege proving data being created from a private key corresponding to a public key assigned to the

service and the result of a unique operation assigned to the client, the method comprising the steps of:

receiving a challenge from the server;

executing the unique operation assigned thereto;

generating a response based on the challenge received from the server, and the result of the unique operation; and

transmitting the response, and the access privilege proving data, to the server.

20. A client that requests services from a server connected to the client via a network, the client comprising:

a unique operation executor for executing a unique operation assigned to the client;

a response generator for generating a response to a challenge received from the server by executing a prescribed calculation using the result of the unique operation; and

a controlling unit, while proving to the server that the client owns the access privilege for the service provided by the server,

for receiving the challenge from the server;

for generating, with the response generator, the response from (a) the challenge, and (b) the result of

the unique operation executed by the unique operation executor; and

for transmitting the created response to the server.

21. A client that requests services from a server connected to the client via a network, comprising:

a portable device connector for connecting to a portable device provided with a unique operation generator for executing unique operations;

a response generator for generating a response to a challenge received from the server by executing a prescribed calculation using the result of the unique operation; and

a controlling unit, while proving to the server that the client owns the access privilege for the service provided by the server,

for receiving the challenge from the server;

for generating, with the response generator, the response from (a) the challenge, and (b) the result of the unique operation executed by the unique operation executor housed in the portable device connected to the portable device connector; and

for transmitting the created response to the server.

* * * * *